POWER MANAGEMENT SYSTEM AND METHOD

Background of the Invention

This invention relates generally to a system and method for managing the power consumption of an electronic device and in particular to a system and method for managing the power consumption of a computer system having one or more electrical components.

With modern electrical systems, it is often desirable to be able to conserve power and reduce the power consumption of the electrical systems. For example, it is desirable to be able to reduce the power consumption of a computer system. For a desktop system that is plugged into an AC outlet, the desirability of power consumption is important. However, for a laptop computer system or any other type of portable computer system that uses battery power or some other limited power source, the desirability of power conservation is critical. In particular, the conservation of power during the operation of the computer system while it is connected to a limited capacity power supply, such as a battery, is critical. The conservation of power leads to longer battery life, which is very desirable.

In general, the power being consumed by an electrical device is equal to $P = CV^2F$ wherein P is the total power, in watts, being consumed by the electrical device, C is the capacitance of the electrical device nodes, V is a voltage being used by the electrical device, and F is the switching frequency of the signal being applied to the electrical circuit. Thus, an electrical device that operates at a high frequency uses more power than the same device at a lower frequency since the transistors and logic gates in the device at the higher frequency will switch more often. With modern electrical devices, the voltage being used by the device (which used to be 5 volts, was 3.3 volts and will soon become 1.8 volts and lower) is somewhat unchangeable since one is typically required to use whatever voltage is currently being required by the semiconductor manufacturers. Therefore, to lower the power being consumed by the electrical device, one can decrease the node capacitance of the electrical device or one can decrease the clock frequency being applied to the electrical device. The capacitance of the

electrical device may be reduced by stopping the operation of one or more portions of the electrical device which reduces the overall switching node capacitance of the device.

The conservation/management of power in a portable electronic device can be achieved by various mechanisms. For example, a portable electronic device typically has one or more different discrete electrical elements/components such as a processor, one or more different memory devices, one or more different electrical buses, one or more input/output interfaces and one or more different peripheral devices which are connected to the buses. The peripheral devices may include a display, such as a liquid crystal display (LCD), a cathode ray tube (CRT) or the like, a persistent storage device, such as a hard disk drive, a removable media storage unit, optical drive, zip drive or the like, one or more input devices, such as a keyboard or mouse and one or more output devices, such as a speaker or an output port. In general, to reduce the power consumption of the portable electronic device, static power management strategies and dynamic power management strategies may be employed. Using static power management techniques, different elements of the portable electronic device may be powered down in order to reduce the power consumption of the portable electronic device by reducing the total capacitance of the portable electronic device. For example, the display of a laptop computer or the display of a portable music device may be powered down during periods when there is no user activity (e.g., the user has not touched the keyboard or mouse for some predetermined period of time). As another example, the hard disk drive of a laptop computer may spin down and stop during the same user inactivity periods. Thus, during periods of inactivity, static power management helps to reduce the power consumption of the system. Using dynamic power management techniques, even during periods of usage, the power consumption of one or more elements of the computer system may be reduced by reducing the effective clock frequency applied to the elements of the computer system for some predetermined amount of time.

The problem and limitation with conventional static and dynamic power management techniques is that one must be able to rapidly return an electrical element to its fastest clock frequency when its clock frequency has been reduced or stopped, otherwise the performance of the portable electronic device may be impaired. In some conventional systems, the clock

frequency applied to one or more elements of the computer system is reduced by reducing the speed (e.g., slowing down) of the well known phase locked loop (PLL) which generates the clock signals for the portable electronic device. Then, the speed of the PLL is increased when it is necessary to increase the clock speed of the electrical element. The problem with that approach is that it takes too long to restore the clock frequency and the performance of the portable electronic device is degraded. In particular, the mechanisms used to sense the need to restore the clock frequency and the time that it takes to communicate that restoration command to the PLL is long, and the time that it takes the PLL to restore its original clock frequency is even longer, such that the total cumulative delay to restore the original clock frequency becomes prohibitively long.

It is desirable to provide a different technique that does not suffer the same problems and limitations as conventional dynamic power management systems. It is also desirable to provide a power management system that incorporates both static and dynamic power management techniques. Thus, it is desirable to provide a power consumption system and method and it is to this end that the present invention is directed.

Summary of the Invention

The power management system and method in accordance with the invention overcomes the limitations and problems with typical power management systems and permits the total power being consumed by a portable electronic device to be minimized so that the battery life of the portable electronic device is maximized. The power management system in accordance with the invention combines one or more power conservation techniques to achieve the best possible power conservation. The power conservation techniques may include static power controls, dynamic power controls, asynchronous buffers and scaleable device clock domains and a flexible clock generator with a software programmable phase locked loop (PLL) that may include one or more different programmable clock policies with programmable clock rates. In a preferred embodiment, there may be four programmable clock policies with four programmable clock rate selectors.

In more detail, the static power control utilizes the means for shutting down/powering down of any unused functional modules at different times that may be accomplished by "gating-off" both logic and clock sources to all of the internal functional blocks. The different functional blocks that may be turned off may include the 24 MHz crystal (X24M) in the clock generator, the 32 kHz crystal (X32K) in the clock generator, the phase locked loop (PLL) in the clock generator, the RTC, the timer (TMR), a audio codec controller (AMC 97), the digital audio exchange interface (SPDIF), the I^2S serial interface bus, the I^2C systems management bus, the universal asynchronous receive/transmit unit (UART), the liquid crystal display device (LCD), the Compact Flash memory card (CF), the Smart Media Card flash memory (SMC), the Multi-Media Card flash memory (MMC), the Sony Memory Stick (MSTK), the general purpose input/output port (GPIO) and interface, the universal serial bus (USB) port interface, the AT Attachment interface (IDE), the Cache Controller (CCH), and either of the processors.

A dynamic power control utilizes the clocking mechanism to reduce power consumption. The asynchronous buffers and scaleable device clock domains for each internal module is provided in order to minimize unnecessary logic switching for each peripheral controllers. The flexible clock generator has a software programmable PLL with prescalar and postscalar to "dial in" or set the appropriate clock frequency to provide just-enough clock speed for the particular task at hand. To use the flexible clock generator, the portable electronic device may include one or more clock policies (four in the preferred embodiment) wherein the clock rate is selected according to the task at hand. The programmable clock rate selectors are provided to automate the dynamic performance boosting for long enough to complete the task using hardware triggered prioritized interrupt services. This provides automated performance scaling to maintain real-time responses and latencies while in different modes at the lowest power consumption.

An hybrid power control utilizes the usual static (spatial) power control mechanisms in a dynamic (temporal) means. Consequently, this technique allows to gate off more portion of a functional module at a much faster rate such that activation of smaller control circuitry would run along the path of data propagation. For example, when a processor wants to access a certain device on a common bus, the bus interface signals are dynamically routed to one and only one

target device for each bus cycles, such that no other device on the same bus would be activated. Unlike the typical implementation of Chip Select which disables other devices from responding, this is implemented by turning-on data path to a single target device, while gating off every other data and control interface signals to every other devices on the same bus. This provides a "virtual disconnect" of the modules that are not in direct line of communication with the processor, such that no power is wasted by the non-participating devices in the high speed internal buses.

Thus, in accordance with the invention, a power management system for a computer system having one or more different components wherein power is dynamically supplied to each component is provided. The power management system comprises a clock generator circuit for generating one or more different clock signals wherein each clock signal has a different predetermined frequency and a clock selector circuit that, based on the task being performed by the computer system, dynamically adjusts the clock signal supplied to each component of the computer system in order to reduce the total power being consumed by the computer system.

In accordance with another aspect of the invention, a power management method for a computer system having one or more different components wherein power is dynamically supplied to each component is provided wherein one or more different clock signals are simultaneously generated wherein each clock signal has a different predetermined frequency and the clock signal supplied to each component of the computer system is dynamically adjusted in order to reduce the total power being consumed by the computer system.

In accordance with yet another aspect of the invention, a flexible clock generator is provided. The clock generator comprises a first oscillator that generates a first clock signal, a second clock oscillator that generates a second clock signal and a programmable clock circuit that generates a third clock signal based on the second clock signal. The clock generator further comprises a clock select circuit that selects one of the first, second and third clock signal that is supplied to a portion of the computer system to provide that portion of the computer system with a predetermined clock signal.

Brief Description of the Drawings

Figure 1 is a diagram illustrating a preferred embodiment of a portable electronic device, such as a portable digital music system, that may include the power management system in accordance with the invention;

Figure 2 is a flowchart illustrating the power management method in accordance with the invention;

Figure 3 is a flowchart illustrating more details of the static power management portion of the power management method;

Figure 4 is a flowchart illustrating more details of the dynamic power management portion of the power management method;

Figure 5 is a diagram illustrating a preferred embodiment of the flexible clock generator in accordance with the invention that may be used to implement the dynamic power management method in accordance with the invention;

Figure 6 is a flowchart illustrating the dynamically programmable power management states in accordance with the invention; and

Figure 7 is a diagram illustrating an example of the power management registers in a preferred embodiment of the invention.

Detailed Description of a Preferred Embodiment

The invention is particularly applicable to a portable digital music device and it is in this context that the invention will be described. It will be appreciated, however, that the power management system and method in accordance with the invention has greater utility, such as to any other electronic device that desirably needs to reduce the power consumption of the device and especially to any limited power source devices, such as battery powered devices, rechargeable battery powered devices and the like, where it is very desirable to reduce the power

consumption of the device. Now, a preferred embodiment of a portable electronic device which is a portable digital music system will be described to provide context for the invention since the invention is applicable to many different electronic systems.

Figure 1 is a diagram illustrating a preferred embodiment of a portable electronic device 60, such as a portable digital music system, that may include the power management system in accordance with the invention. In this embodiment, the device shown has been implemented as a two processor system of a chip, but the power management system is applicable to any type of electronic device. The system may also include a cross bar multipath memory controller 62 and a cross bar multipath peripheral controller 64 which are described in more detail in copending patent application serial number 09/XXX,XXX filed on XXXXXXXX and entitled "Cross Bar Multipath Resource Controller System and Method" which is owned by the same assignee as the present invention and which is incorporated herein by reference.

As shown, the multiple processor system 60 may include a host processor 66 which may preferably be a reduced instruction set (RISC) ARM core made by ARM Inc and a coprocessor core 68 that operate in a cooperative manner to complete tasks as described above. In the preferred embodiment, there may also be a hardware accelerator engine 70 as shown. A software DMA engine 71 in this preferred embodiment may be executed by the coprocessor core 68. The software DMA engine is described in more detail in copending patent application serial number 09/XXX,XXX filed on XXXXXXXX and entitled "Software Direct Memory Access Engine for Multiple Processor Systems" which is owned by the same assignee as the present invention and which is incorporated herein by reference.

In more detail, the host processor, the coprocessor and the hardware accelerator engine are all connected to the multipath memory controller 62 and the multipath peripheral controller 64 as shown which permit the host processor and the coprocessor to access each shared resource using its own bus. To control access to the shared resources connected to the multipath memory controller and the multipath peripheral controller, the system 60 may include a semaphore unit 72 which permits the two processors 66, 68 to communicate with each other and control the

access to the shared resources. The details of the semaphore unit is described in more detail in copending US patent application number XX/XXX,XXX filed on XXXX,XX 2001 titled "Multiprocessor Communications System and Method", owned by the same assignee as the present invention and incorporated herein by reference. The semaphore unit permits the processors to negotiate for the access to the shared resources as described above, but then, due to the multipath controllers 62, 64, permits the processors to access the resources over its own bus that is part of the controllers. To control the timing of the controllers 62, 64, a timer/clock 74 is connected to each controller 62, 64.

Both the memory controller 62 and the peripheral controller 64 are then in turn connected to one or more resources that are shared by the processors. For example, the memory controller 62 in this preferred embodiment is connected to a host instruction memory 76 that is typically accessed by the host processor 66, a ping buffer 78 that may be accessed by each processor as needed, a pong buffer 79 that may be accessed by each processor as needed and a coprocessor instruction memory 80 which is typically accessed by the coprocessor 68. Due to a priority scheme and the cross bar architecture, the host processor may always have priority access to its instruction memory 76 and the coprocessor may always have priority access to its instruction memory 80 since the two processors each have separate buses connected to each resource. The memory controller 62 may also be connected to a cache memory 82, which is a well known 4-way 4 kB set associative cache in the preferred embodiment, a flash memory interface 84 for connecting to an external flash memory and an external synchronous dynamic random access memory (SDRAM) interface 86 with the various necessary signals, such as RAS, CAS, WE, OE and CS, to interface to a typical well known SDRAM.

The peripheral multipath controller, which operates in a similar manner to the memory controller in that each processor may access different shared resources simultaneously, may have one or more peripherals connected to it. In the preferred embodiment, the peripheral controller may be connected to a universal serial bus (USB) interface 88 that in turn connects to a USB device or host, a universal asynchronous receiver/transmitter (UART) interface 90 that in turn connects to communication port (COM) hosts, a TAP/embedded ICE controller 92, an EIDE-

CD/CF controller 94 to interface to hard disk drives or CD drives, a key matrix controller 96 that connects to a user input keyboard, an audio-codec controller 98 that connects to an audio coder/decoder (codec), an liquid crystal display (LCD) display controller 100 that connects to a LCD display, a smartcard controller 102 for connecting to a well known smart card and an input/output (I/O) expansion port 104 that connects to one or more different input/output devices. As with the memory controller, the peripheral controller provides access for each processor to each shared resource. Now, the power management system that is incorporated into the above portable electronic system will be described.

Figure 2 is a flowchart illustrating a power management method 110 in accordance with the invention that reduces the total power consumption of the above system. Broadly, the power management method may include the steps of static power management 112 and dynamic power management 114 as shown in Figure 2 that are performed continuously during the operation of the portable electronic device. More details of the power management states in accordance with the invention are described below with reference to Figure 6. The power management method shown in Figure 2 is somewhat simplistic in that both the static and dynamic power management steps incorporate one or more sub-steps which will now be described in more detail.

In general, the goal of power management is to reduce the total power consumption of the portable electronic device. In general, the goal is achieved by only providing power to the devices that need to be in operation at any particular time (e.g., static power management) or by reducing the power consumed by particular devices during particular operations (e.g., dynamic power management). To facilitate this reduction in power consumption, there are a number of desirable options for implementing the reduction. First, there may be dynamic usage which mostly applies to the host processor and coprocessor devices wherein the processor devices should always be placed in a static/halted state when there is "nothing to do". For the host processor, that could involve halting during the RTXC "null task". For the coprocessor, it may be managed by the decoder loop. In a preferred embodiment, each software component in the system may be more-or-less event-driven so that there is no busy-waiting state which wastes power.

On some of the devices within the system, there are logic blocks that do their own "dynamic clocking" to reduce power consumption. For example, the IDE controller 94 uses little or no power until it is accessed. For other device that do not have "dynamic clocking", it may be desirable to provide one or more of the following techniques. First, since each customer will not use all the hardware all of the time, a mechanism for disabling (declocking) hardware that is not used in a particular configuration is desirable and for configuring the firmware to not access it during that time is also desirable. Second, each device that is used intermittently may have an Open/Close driver model wherein when a device is opened, its hardware is powered on and when it is closed the hardware is powered off. Third, some devices have media that is removed or inserted into it or a cable that plugs into it, such as a ZIP drive, a smartcard slot and the like. When there is no media present, the device should be powered down and when the media is inserted, the device is powered up. Fourth, timeouts may be used which are good for devices associated with user input/output (I/O) such as the back-light, the LCD, spinning media, and the system as a whole. Since the timeouts are related to user activity, they have to be configurable as to duration and as to which activities reset them. In general, the goal of the power management system is to cope with wide ranges of performance at a lowest level power consumption by utilizing automated power conservation techniques to achieve "Just Enough Performance for Just Enough Time."

In more detail, the power management system may provide the static controls as well as the dynamic controls. A static power management method 120 in accordance with the invention are shown in Figure 3. The static power management method begins in that one of the processors may be executing a background task that checks if a device in the system is being used in step 122. If an element is not currently being used, the system may gate off the logic and clock of the device in step 124. The system may then check if the device is needed in step 126 and reactivate the device in step 128 when needed. In this manner, the static power management techniques are used to power down any device that is not being used at the current time. In accordance with a preferred embodiment of the invention, the power may be removed from one or more of the following devices/modules including the 24 MHz crystal (X24M) in the clock

generator, the 32 kHz crystal (X32K) in the clock generator, the phase locked loop (PLL) in the clock generator, the RTC, the timer (TMR), a audio codec controller (AMC 97), the SPD interface (SPDIF), the I^2S bus, the I^2C bus, the universal asynchronous receive/transmit unit (UART), the liquid crystal display device (LCD), the CF, the SMC, the MMC, the MSTK, the general purpose input/output port (GPIO) and interface, the universal serial bus (USB) port and interface, the IDE port and interface, the CCH and either of the processors. The registers associated with the static power management will be described below with reference to Figure 7.

Figure 4 illustrates a dynamic power management method 130 in accordance with the invention. The dynamic power management may include the system determining if the system is in a different operating mode in step 132 as described below with reference to Figure 6. If the system has entered into a different mode of operation, then the system may dynamically change the power management mode of the system in step 134. In this manner, the power applied to device within the system may be dynamically changed based on the current operating mode of the system. In a preferred embodiment, dynamic power controls for all "active controllers" (e.g., active Controllers are those devices that are being used for communication, either with external device or the internal CPU) are designed into the chip-select decode logic such that only the selected controller's combinatorial logic would be active for communicating with the main processor. This provides a virtual disconnect of the modules from the high speed internal buses. In a preferred embodiment, interrupt driven algorithms may be used whenever possible instead of polling-mode algorithms both of which are well known.

In more detail, each typical chip may include address inputs, control inputs, data inputs and data outputs with a chip select wherein the chip select is used to typically gate off the control inputs to the chip. To implement the virtual disconnect in accordance with the invention, the chip select signal may also be used to gate off the address inputs, control inputs, data inputs and data outputs so that the chip/device is disconnected from the bus and therefore does not consume power. To implement the virtual disconnect in accordance with the invention, the repeater buffers which are typically present in modern design may be used.

To further enhance the power management capabilities, the system may include asynchronous buffers and scaleable device clock domains for each internal modules to minimize unnecessary logic switching for each peripheral controllers. This provides "Just enough performance" to support individual levels of activity for each individual controllers.

In more detail, each device controller should be optimized for the lowest possible clock operation. For example, when the system is communicating with a serial UART device, the system should be able to slow our clocks to our CPU so that the CPU can talk to other device at it's slower baud rate, instead of running at a fast clock and waiting for the slow device to respond. This becomes much more significant when the processor must wait for user inputs. Therefore, if we drop the CPU clock rate to keep pace with human interaction, we should be able to save some power. On the other extreme, if the CPU has to encode music, it needs to run at top speed to do the complex operations. This full speed operation consumes a lot of power, but this is the power consumed for doing real work.

The system may also include a flexible clock generator, as described in more detail below with reference to Figure 5, with a programmable PLL with Prescalar and Postscalar to select the appropriate clock frequencies for "Just Enough Processing Performance". The hardware registers associated with controlling the flexible clock generator will be described below with reference to Figure 7. The system may also include, as part of the dynamic power management, one or more programmable clock states and policies (four in the preferred embodiment) with one or more corresponding programmable clock rate selectors (four in the preferred embodiment) that are provided to automate the dynamic performance boosting for "Just Enough Time" via hardware triggered prioritized interrupt services. The hardware registers associated with the programmable clock states is described below with reference to Figure 7 and the programmable clock states are described below with reference to Figure 6. The system also has a symmetric multiprocessor architecture that is provided to balance the work-load between two processors to minimize overall clock frequency and thus the overall power consumption. Now, the flexible clock generator in accordance with the invention will be described.

Figure 5 is a diagram illustrating a preferred embodiment of a flexible clock generator 140 in accordance with the invention that may be used to implement the dynamic power management method in accordance with the invention. To provide a flexible clock signal, the flexible clock generator may include a first crystal oscillator 142 and a second crystal oscillator 144 as shown. The first crystal oscillator may be connected to a 32.768 KHz crystal 146 and may generate a 32 kHz clock signal that is output to an oscillator pin 150 and through a buffer 152. The second oscillator 144 may be connected to a 24.576 MHz crystal 148 and may generate a 24 MHz clock signal that is output to an oscillator pin 154 and through a buffer 156. Thus, a 32 KHz clock signal and a 24 MHz clock signal are generated.

The flexible clock generator 140 may also include a 16 bit PNG LFSR 158 which is a random number generator (a preferred random number generator is described in co-pending patent application number 09/XXX,XXX. filed XXXX and titled "XXX" which is incorporated herein by reference), a usTimer 160 that generates a usTicks signal (this is a micro-second time pulse generator for various timing control) and a real-time clock and calendar (RTC) element 162. The real-time clock and calendar element may receive the 32 KHz clock signal as an input that is fed into a 5-bit prescalar 164, a 6 bit prescalar 166, a 6 bit prescalar 168 and a 15 bit prescalar 170 that are used to generate the real-time time and date for the system. In particular, one prescalar generates a day (Days) output, one prescalar generates a hours (Hrs) output, one generates a minutes (Min) output and one generates a seconds (Sec) output as is well known.

The flexible clock generator 140 may further include an 8 bit postscalar 172, an 8 bit prescalar 174 and a clock select mutliplexer 176. The outputs from the postscalar and prescalar 172, 174 are input into a well known phase locked loop (PLL) 178. The output from the PLL is fed into a 4 bit RunCLK unit 180 and a 4 bit BurstCLK unit 182 wherein the RunCLK unit generates a 33 MHz clock signal (33M) and the BurstCLK unit generates a 66 MHZ clock signal (66M). The 33 MHz clock signal (33M), the 66 MHz clock signal (66M), the 24 MHz clock signal (24M or MainCLK) and the 32 kHz clock signal (32K or StbyCLK) are all fed into the clock select multiplexer (CLK SLCT) 176 that selects one of the clock signals to output. Thus, as shown, the clock select multiplexer may output one or more different frequency clock signals

including a 32 kHz signal, a 24 MHz signal, a SysCLK signal, a CpuCLK signal and a CopCLK signal.

To control which clock signal is output at a particular time, there may be a clock policy unit (CLK POLICY) 184 that provides signal to the multiplexer to select a particular clock signal. The clock policy unit is in turn driven by a clock state unit (CLK STATE) 186 that may be a state machine. The clock state unit may be driven by various interrupt signals such as an FIQ signal, an IRQ signal and software commands (SW) as shown. In operation, the interrupt signals or software commands may indicate a new operational state of the system and the state machine may transition to the new state which points to a particular location in a hardware register as described below. The state machine may then generate the appropriate signals based on the hardware register location(s) so that the clock policy unit may generate the appropriate multiplexer control signals. For example, the system may enter an idle state and the state machine may transition to the IDLE state. The hardware register corresponding to the IDLE state may indicate that the 32 KHz clock signal for certain devices in the system and the 24 MHz signal for other devices should be output and the clock policy unit may generate the appropriate control signals to the multiplexer.

In more detail, the variable rate clocks are utilized by the frequency multiplier PLL and the Burst/Run/Cruse/Standby clock arbitration selector 176. The fixed rate 24MHz clock signal is used for a baud rate counter and timing sequence applications, while the 32KHz clock is used to generate Real Time Clock and Calendar (RTC) during normal operations as described above. The programmable rate SysCLK shall be used to enable synchronous operation of the system resources. The CpuCLK and CopCLK shall be used for the primary and the secondary processor core operations. When dynamic rate shifting is enabled, these clocks (SysCLK, CpuCLK and CopCLK) shall be automatically and independently shifted to a higher frequency when a Fast Interrupt Request (FIQ) is activated. For example, while the processor is running at StandBy mode (32KHz), the FIQ could activate the 24MHz burst mode. As another example, while the processor is running at a Normal mode clock frequency (~40MHz), the activation of the FIQ could switch-on the Burst Clock frequency (~80MHz). Consequently, the hardware provided the

shortest time to wakeup with bursting clocks, the Firmware (IRQ/FIQ Service Routine) provided the shortest time to sleeping at suspended clocks for the maximum power savings. By default, all clocks may run at the top clock frequency during a Power-On Cold Reset. When the CpuCLK and CopCLK are in a Stop Clock Mode, activation of IRQ may switch-on the clock frequency for the 32KHz clock. Now, the programmable power management states in accordance with the invention will be described in more detail.

Figure 6 is a flowchart illustrating the dynamically programmable power management states 200 in accordance with the invention. As shown, there may be a BUSY state 202, an IDLE state 204, a DEAD state 206 and a SLEEP state 208 and the power management state can transition between the states as shown. The different power management states may be thought of as power management "macros" for balancing system responsiveness and power usage. The states focus on management of the host processor and the coprocessor. For each system state, other devices are still expected to be power managed on an individual/independent basis as set forth below. In more detail, individual devices interrupt the Supervisor for a certain requests where the Supervisor assigns appropriate tasks to satisfy the request in a timely manner. Consequently, the Supervisory service routine determines the nature of the service required and schedules the resource and task required. Now, an example of the possible states, the set-ups for the states and the transitions to other states in a preferred embodiment will be described.

Busy State

In this state, the system is performing a performance-intensive task. In the preferred digital music device, those performance-intensive tasks may include digital music playback, music recording, music download, music upload, music data transfer, firmware updates, etc.

Setup: The processor's cores are clocked off the PLL at high speeds (60-70MHz or whatever is determined to be adequate), the IRQs and FIQs are run at same rate, other devices are powered as needed and idle tasks just halt the host processor and the coprocessor.

Busy To Idle Transition: Since the system has a firmware "supervisor" who can potentially know when intensive tasks are in progress, we can have the supervisor manage transitions into and out of the Busy state based on what commands come in from the user or from other communications. Whenever an intensive task finishes, the supervisor switches the system back to the Idle state.

Idle State

In this state, the system is either waiting for the user to do something "significant" (e.g., strike a key or the like) or is connected to an active communications link and is waiting for a command (download, upload, etc.) or is waiting to time out. There are at least two potential "sub-modes". In one sub-mode, if the user it not around, the display can time out, but the processor needs to stay up so we can communicate. In a second sub-mode, if the user is around, then essentially only the screen and keypad/touch support needs to be kept running which is a "lights are on but no one is home" mode.

Setup: In the mode, the coprocessor is off, the PLL is off and the host processor is clocked at 32KHz. In addition, the IRQs and FIQs are run at 24MHz, the auto bit is set so that the clock jumps to 24MHz on an IRQ or FIQ signal and other devices are powered as needed.

Idle to Busy Transition: When the supervisor starts an intensive task as described above, it transitions the system to the Busy state.

Idle to Sleep Transition: If there is no activity, either by the user or from the communication ports, for a period of time, the system bookmarks its current activity to flash memory (stores its data) and goes to the Sleep state. The bookmark, for example, may contain information like: "paused playing Moonlight Sonata at 2:34:104 from file "Moonlight" on CF card serial number 1029384756".

Idle to IRQ/FIQ Transitions: When an IRQ or FIQ interrupt signal occurs, the host processor may switch to a 24MHz clock. The CPU may remain at 24MHz until it drops into the

RTXC null task and then the clock is cycled back down to 32KHz. This may allow screen updates and so forth to run quickly enough.

Sleep (Hibernate) State:

In this state, the user has turned the system "off" or the system has timed out because of user and communications inactivity or batteries have been drained. In this mode, general purpose devices have to keep the RAM active (to avoid losing data) when they sleep, or they have a "hibernate" mode where they save the RAM contents to a hard disk or whatever. In accordance with the invention, a Hibernate mode may be implemented by bookmarking (see description above) what we were doing to flash and power everything off except the RTC and any wakeup circuits.

Setup: In this mode, everything is turned off except the RTC and active wakeup circuits, if any.

Sleep to Idle Transition: In response to an "on" command, the system transitions into the Idle state, gets the bookmark from flash, and then decides whether it needs to go to Busy state or not. It is also possible to transition directly into the Busy state and then drop into the Idle state. However, there is a greater risk of killing off almost-dead batteries before you have a chance to measure them.

Dead State

In this mode, the power has failed. The bookmarked system state should still be safely stored in flash, but the time and date will need to be set upon restart.

Dead to Idle Transition: The transition is the same as the Sleep to Idle transition, except that the time and date will need to be set and they may not be a bookmarked system state in flash (if this was the first time the unit was powered up). In the alternative, there may be a default bookmark that will be placed in flash during manufacturing.

Global Transitions

Busy or Idle to Sleep Transition: On an "off" command or if the system runs out of power from the battery, the system may go into the Sleep state. Since there are some complication that result when the user tries to put the system into a sleep mode while it is busy writing flash, communicating or whatever, the system may prompt the user to wait for some period of time or cancel some pending operation before sleeping. The system may also ensure that the system operations are complete before shutting the screen off so that the user does not inadvertently swap batteries or some other action, which could damage the system. Now, an example of the power management registers in accordance with a preferred embodiment of the invention will be described.

Figure 7 is a diagram illustrating an example of the power management registers 220 in a preferred embodiment of the invention. For static power management, the clock enable registers 222, 224 are shown (Clock Enable Registers $CF00:5000 and $CF00:5004). The different clock registers to select a particular clock frequency from the flexible clock generator are registers 226, 228 ($CF00:5018 and $CF00:501C). In addition, the preferred system includes the four programmable clock policies with four programmable clock rate selectors are provided to automate the dynamic performance boosting for "Just Enough Time" via hardware triggered prioritized interrupt services. This provides an automated performance scaling to maintain real-time responses and deterministic latencies while "Running" or "Idling" at a lowest clock rate for the maximum power conservation. The registers 230, 232, 234 that control these clocks are shown. ($CF00:5008, $CF00:500C, $CF00:5010).

Some experimental results associated with the above described power management system and method are now provided. In particular, listed below are two experimental power consumption measurement to be used as our Playback performance Goal. To perform the test, the test program was written to enable Cache then Idle Loop on IRAM. From the Multi-ICE debugger, various modules were then disabled as indicated while observing the power

consumption. First, the power consumption of a single processor at 49MHz was determined. The results for the single processor are:

a. 104mA at 2.5V core, Cache=ON, all Dev.CLKs=OFF, all Dev.RSTs=OFF, JTAG=ON. (260mW)
b. 103mA at 2.5V core, Cache=ON, all Dev.CLKs=OFF, all Dev.RSTs=ON, JTAG=ON. (258mW)
c. 76mA at 2.5V core, Cache=OFF, all Dev.CLKs=OFF, all Dev.RSTs=ON, JTAG=ON. (190mW)
d. 82mA at 2.5V core, Cache=ON, all Dev.CLKs=OFF, all Dev.RSTs=ON, JTAG=OFF. (205mW)
>> 27mA for Cache Controller (67mW)
>> 21mA for JTAG Debug Controller (52mW)
>>>> 55mA for CPU-IRAM (137mW)

The results for dual processor that are each running at 24.578MHz are:

a. 68mA at 2.5V core, Cache=ON, Dev.CLKs=MMC-Play, JTAG=ON (170mW)
b. 54mA at 2.5V core, Cache=OFF, Dev.CLKs=MMC-Play, JTAG=ON. (135mW)
c. 58mA at 2.5V core, Cache=ON, all Dev.CLKs=MMC-Play, JTAG=OFF. (145mW)
>> 14mA for Cache Controller (35mW)
>> 16mA for JTAG Debug Controller (40mW)
>>>> 38mA for CPU-IRAM-MMC-AC97 (95mW)

From the above experimental results, it was determined that, for the same amount of performance (49 million instructions per second (MIPS)), the use of single processor running at 49MHz consumes more power than utilizing two processors at half the clock speed (24MHz). this results is due to the clock distribution circuit for the support logics (IRAM, BUSES, CACHES, etc) having to idle at a higher clock rate. In more detail, although the processor provides the same amount of work and thus the same power consumption (22mW at 49MHz versus 21mW at 24MHz), the clock distribution losses from the synchronous Flops are significant.

The experimental results also indicated that the JTAG Debug Controller should be turned off for the production units because it utilizes considerable power when not disabled. The experimental results also indicated that, if at all possible, do not activate the control logics when the module is not the target for the bus cycle. It appears that the system may be able to save some more power if we "pre-determine" our target address (one of four IRAM blocks,Cacheable Cycles, Memory Cycles, and Peripheral Cycles) and activate the Address/Data/Control/Clock to the target devices only.

The experimental results also indicated that when the external IO interface is not being used, the outputs may be driven to the inactive state of the line according to its pull-up or pull-down configurations. If the output is tristateable, tri-state the un-used pins. If the bidirectional pin is not used, drive the pin to it's pulled-up or pulled-down state before tristating the output. Now, the results of the system operation with only the static power management and without any power management is shown.

Power Use With Static Power Controls

	TOTAL	CORE	I/O DEVICES	AUDIO AMP
SLEEP (32 KHz)	(16 mA 1.5 V) 0.024W DCIN	(1mA 2.5V)** 3mW	(5mA, 3.3V)** 16 mW	(1mA, 5V)** 5mW
IDLE (24 MHz)	(148mA 1.5V) 0.222W DCIN 19.3%	(32mA, 2.5V) 80mW 11.6%	(34mA, 3.3V) 112mW 47.9%	(6mA, 5V)* 30mW 13.0%
PLAY (49 MHz)	(558mA 1.5V) 0.837W DCIN 70.1%	(161mA, 2.5V) 403mW 56.2%	(62mA, 3.3V) 205 mW 82.0%	(46mA, 5V) 230mW 100%
REC (66 MHz)	(664mA 1.5V) 0.996W DCIN 74.9%	(222mA, 2.5V) 555 mW 76.8%	(64mA, 3.3V) 211mW 56.1%	(46mA, 5V) 230 mW 100%

Without the power management controls, the power consumption levels are:

	TOTAL	CORE	I/O DEVICES	AUDIO AMP
Power On	(726 mA 1.5 V) 1.089W DCIN	(275mA 2.5V)** 687mW	(52mA, 3.3V)** 172 mW	(46mA, 5V)** 230mW
IDLE (66 MHz)	(768mA 1.5V) 1.151W DCIN	(275mA 2.5V)** 687mW	(71mA, 3.3V) 234mW	(46mA, 5V)* 230mW
PLAY (66 MHz)	(795mA 1.5V) 1.193W DCIN	(285mA, 2.5V) 713mW	(76mA, 3.3V) 250 mW	(46mA, 5V) 230mW
REC (66 MHz)	(886mA 1.5V) 1.329W DCIN	(289mA, 2.5V) 723 mW	(114mA, 3.3V) 376mW	(46mA, 5V) 230 mW

In a preferred embodiment, the features of the system may include:

1. Record and Playback using ROM, SRAM, MMC, Static LCD with 1 second refresh or blinking Icon.

2. Record from Line-In (CD) at 32KSS with 64kbps, 128kbps, or 256kbps compression rate.

3. Record from Line-In (FM) at 22KSS with 32kbps, 64kbps, or 128kbps compression rate.

4. Record from Mic-In (Voice) at 8KSS with 32kbps, or 64kbps compression rate.

The power control settings below (as shown in a Reference Board) are described. The SLEEP mode is when unit is "OFF" while power is still supplied. The IDLE mode is when the unit is in Standby or in the process of accepting user commands. The PLAY mode is when the unit is decoding and playing back MP3 file. The REC mode is when the unit is encoding MP3 and recording to a file. The Control Setting could be done in the order specified, but standard programming precaution should be utilized to avoid system hang-up. Please refer to GPIO assignment below for the associated power control bits.

SLEEP: Wait for WakeUp Events @ 32KHz SysClock (Buttons or Card Detects)
AudioAmp=OFF, AC97=PD, Display=OFF, BackLight=OFF
$CF00:5000 = 0008h (ClockOn = RTC, ClockOff= all others)
$CF00:5004 = 0001h (ClockOn = GPIO, ClockOff = all others)
$CF00:5008 = 0001h (Static CPU clock setting at IDLE Clock rate)
$CF00:500C = 0000h (Set Clock Policy as 32K/32K/32K/32K settings)
$CF00:5010 = 2000h (TurnOff=PLL, 24M. TurnOn=32K)
$CF00:501C = 000Ch (Postscale = 12x) PLLCLK = 49MHz (off)
$CF00:5018 = 0006h (Prescale = 1/6) RefCLK = 4MHz (off)
$CF00:4000 = 2044h (ROM access time with 70ns Flash)
$CF00:4008 = 2022h (RAM access time with 12ns SRAM)

IDLE: Wait for Control Events @ 32KHz SysClock (Buttons, Card, or Comm Detects)
AudioAmp=OFF, AC97=PD, Display=ON, BackLight=OFF
$CF00:5000 = 0018h (ClockOn = TMR and RTC, ClockOff = all others)
$CF00:5004 = 0031h (ClockOn = CCH and GPIO, ClockOff = all others)
$CF00:5008 = 0081h (Dynamic CPU clock setting at IDLE Clock rate)
$CF00:500C = 0054h (Set Clock Policy as 24M/24M/24M/32K settings)
$CF00:5010 = 6000h (TurnOff=PLL. TurnOn=24M,32K)
$CF00:501C = 000Ch (Postscale = 12x) PLLCLK = 49MHz (off)

$CF00:5018 = 0006h (Prescale = 1/6) RefCLK = 4MHz (on)
$CF00:4000 = 2044h (ROM access time with 70ns Flash)
$CF00:4008 = 2022h (RAM access time with 12ns SRAM)

PLAY: Play from MMC @ 49MHz SysClock? (We need to optimize it later)

AudioAmp=ON, AC97=ON, Display=StaticON (blink Icon), BackLight=OFF
$CF00:5000 = 403Ah (ClockOn = MMC, AC97, TMR, RTC, C.24M. ClockOff = all else)
$CF00:5004 = 0031h (Clock_Off = IDE, USB)
$CF00:5008 = 0084h (Dynamic CPU clock setting at MAIN Clock rate)
$CF00:500C = 00E4h (Set Clock Policy as PLL/PLL/24M/32K settings)
$CF00:5010 = E000h (TurnOn=PLL, 24M,32K. Set BurstClk = RunClk = PLLCLK)
$CF00:501C = 000Ch (Postscale = 12x) PLLCLK = 49MHz (on)
$CF00:5018 = 0006h (Prescale = 1/6) RefCLK = 4MHz (on)
$CF00:4000 = 2044h (ROM access time with 70ns Flash)
$CF00:4008 = 2022h (RAM access time with 12ns SRAM)

REC: Record to MMC @ 66Mhz SysClock

AudioAmp=ON, AC97=ON, Display=StaticON (blink Icon), BackLight=OFF
$CF00:5000 = 403Ch (Clock_Off = Mstk, SMC, LCDC, UART, I2C, I2S, SPDIF, C.EXT, C.32K)
$CF00:5004 = 0031h (Clock_Off = IDE, USB)
$CF00:5008 = 0008h (Static CPU clock rate using Burst Clock policy)
$CF00:500C = 00E4h (Set Clock Policy as PLL/PLL/24M/32K settings)
$CF00:5010 = E000h (TurnOn=PLL, 24M,32K. Set BurstClk = RunClk = PLLCLK)
$CF00:501C = 0010h (Postscale = 16x) SysCLK = 66MHz (on)
$CF00:5018 = 0006h (Prescale = 1/6) RefCLK = 4MHz (on)
$CF00:4000 = 2044h (ROM access time with 70ns Flash)
$CF00:4008 = 2022h (RAM access time with 12ns SRAM)

While the foregoing has been with reference to a particular embodiment of the invention, it will be appreciated by those skilled in the art that changes in this embodiment may be made without departing from the principles and spirit of the invention, the scope of which is defined by the appended claims.